Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dragan Dubravac

Spletna aplikacija za napoved pretoka prometa

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Branko Šter

Ljubljana, 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Spletna aplikacija za napoved pretoka prometa:

Izdelajte aplikacijo za napoved pretoka prometa in prometnih zastojev na podan datum in uro na osnovi podatkov iz preteklosti. Opišite model, ki je uporabljen za napovedovanje pretoka prometa. Opišite aplikacijo in njene lastnosti ter navedite nekaj primerov njenega delovanja pri napovedi pretoka in zastojev v prometu.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Dragan Dubravac, z vpisno številko **63080218**, sem avtor diplomskega dela z naslovom:

Spletna aplikacija za napoved pretoka prometa

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Branka Štera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 7. oktober 2015 Podpis avtorja:

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	programski vmesnik
IDE	Integrated Development Environment	razvojno okolje
GUI	Graphical User Interface	grafični uporabniški vmesnik
...	...	...

Povzetek

Naslov: Spletna aplikacija za napoved pretoka prometa

Motnje v prometu na avtocestah ponavadi povzročijo zgoščen pretok prometa ali pa celo zastoj. Glavni cilj diplomskega dela je izdelava aplikacije za napoved pretoka prometa in prometnih zastojev na podan datum in uro s pomočjo podatkov iz preteklosti. Z uporabo povprečne hitrosti lahko, s pomočjo podatkov števnih mest, napovemo pretok prometa in razvoj zastojev.
Aplikacija prikaže graf, ki predstavlja število avtomobilov na dani števni lokaciji, ter upošteva pretok posamezne ceste in napove zastoje na cestah. Izdelana aplikacija je namenjena uporabnikom, ki želijo predvideti prometne zastoje. Za izdelavo aplikacije smo uporabili programski jezik Java, da lahko dosežemo neodvisnost od platforme, na kateri bomo aplikacijo uporabljali, ter možnost preproste objave na spletu.

Ključne besede: Java, promet, zastoj, spletna aplikacija.

Abstract

Title: Web application for traffic flow forecasting

Disturbances in traffic on highways usually lead to an increase in traffic flow density or even traffic jams. The main goal of the diploma thesis is to produce an application that will predict traffic flow and traffic jams on any given date and time based on traffic flow in the past. By using an estimated desired speed the traffic information gained from flow counters we can forecast the evolution of traffic jams.
The application presents a graph that represents the number of vehicles on a given road section and forecasts potential traffic jams. The application is meant for users who wish to predict traffic jams. For the production of the application we have used the Java programing language for its cross-platform capability and ease of web deployment.

Keywords: Java, traffic, traffic jam, web application.

Poglavje 1

Uvod

Napovedovanje in s tem modeliranje prometa, se nanaša na interpretacijo informacij o prometu, z namenom razumevanja in razvoja optimalnega prometnega omrežja. Napovedovanje prometa se ponavadi začne z zbiranjem podatkov o trenutnem stanju prometa. Ti podatki se nato združijo z ostalimi znanimi podatki, kot so količina prebivalstva, potni stroški, vremenske razmere ipd., z namenom razvoja zahtev modela. Prometne napovedi se uporabljajo za več namenov: od načrtovanja transportnih poti do ocenjevanja zmogljivosti prometne infrastrukture.

Cilj te diplomske naloge je bil razviti aplikacijo, ki napove količino prometa na podan datum. S to informacijo nato izračunamo, koliko časa bo trajal zastoj, če količina prometa presega kapaciteto ceste.

Za doseganje cilja napovedovanja prometa smo uporabili neparametrični model, ki uporablja podatke s številnih mest iz preteklosti za napovedovanje stanja prometa v prihodnosti. S tem pristopom smo zmanjšali število predpostavk o naših podatkih, kar nam razširi uporabnost aplikacije ter jo naredi bolj robustno. Model na podlagi informacij o prometu, pogostih destinacijah, gostoto prebivalstva ..., poizkusi napovedati promet na podan dan. Z našim pristopom zmanjšamo število potrebnih informacij in s tem parametrov, kar nam omogoča večjo robustnost končnega programa. V kombinaciji z izdelavo programa v Javi smo dosegli platformno neodvisnost ter preprostost uvajanja

programa na spletu.

Za orodja smo si izbrali programski jezik Java zaradi prenosljivosti končnega programa in razvojno okolje NetBeans zaradi močnega in preprostega orodja za oblikovanje grafičnih uporabniških vmesnikov.

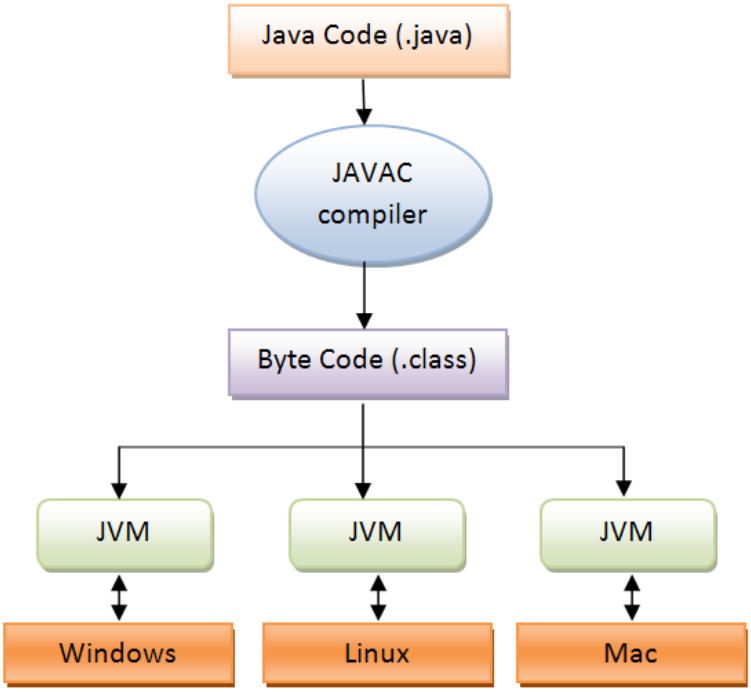
Poglavje 2

Predstavitev uporabljenih orodij in tehnologij

2.1 Java

Java je objektno usmerjen, prenosljiv programski jezik, ki so ga razvili v podjetju Sun v začetku 90-ih. Razvit je bil kot zamenjava za programski jezik C++, zato je sintaksa Jave podobna jezikoma C in C++. Java odpravi določene nizko-nivojske konstrukte (npr. kazalce) in ima zelo preprost način upravljanja s pomnilnikom, kjer se vsak objekt shrani v kopico in vse spremenljivke objektov so sklici na te objekte. Upravljanje s pomnilnikom je vodeno s samodejnim sproščanjem pomnilnika (angl. Automatic Garbage Collector), kar olajša delo programerja, ki mu ni potrebno skrbeti za brisanje podatkov v pomnilniku, ki jih ne potrebuje več.

V nasprotju s konvencionalnimi programskimi jeziki, ki so v splošnem narejeni tako, da izvorno kodo s prevajalnikom prevedejo v strojno kodo ali pa se izvorna koda tolmači že med izvajanjem, je pri Javi ta postopek drugačen. Izvorno kodo prevedemo v vmesno kodo (angl. bytecode, slika 2.1), ki jo nato poganja Javin navidezni stroj (angl. Java Virtual Machine). S tem dosežemo neodvisnost od platforme, kar pomeni, da se lahko programi v Javi izvajajo brez ponovnega prevajanja na vseh platformah.



Slika 2.1: Vmesna koda se z pomočjo JVM lahko uporabi na vseh platformah.

Programerji so si pri razvoju Jave zastavili nekaj glavnih ciljev:

- Objektno usmerjeno programiranje
- Neodvisnost od platforme
- Vgrajena podpora za mrežno in medmrežno komunikacijo
- Enostavnost uporabe

2.1.1 Java Swing

Swing je knjižnica z razredi za razvoj grafičnih uporabniških vmesnikov v Javi. Knjižnica je implementirana v celoti v Javi in zaradi tega ponuja enoten izgled komponent, ne glede na platformo. Prednosti Swing-a so:

- lahka izvedba - gradniki niso vezani na izgled sorodnih gradnikov v sistemu, kjer se program izvaja
- zamenljiv videz uporabniškega vmesnika - gradnikom lahko spreminjamo videz neodvisno od platforme
- podpora "povleci in spusti"
- dodaten nabor gradnikov, kot so drevesa, izbirnik datotek,...
- zmogljivejša zasnova gradnikov - omogoča prilagajanje gradnikov in ustvarjanje lastnih gradnikov

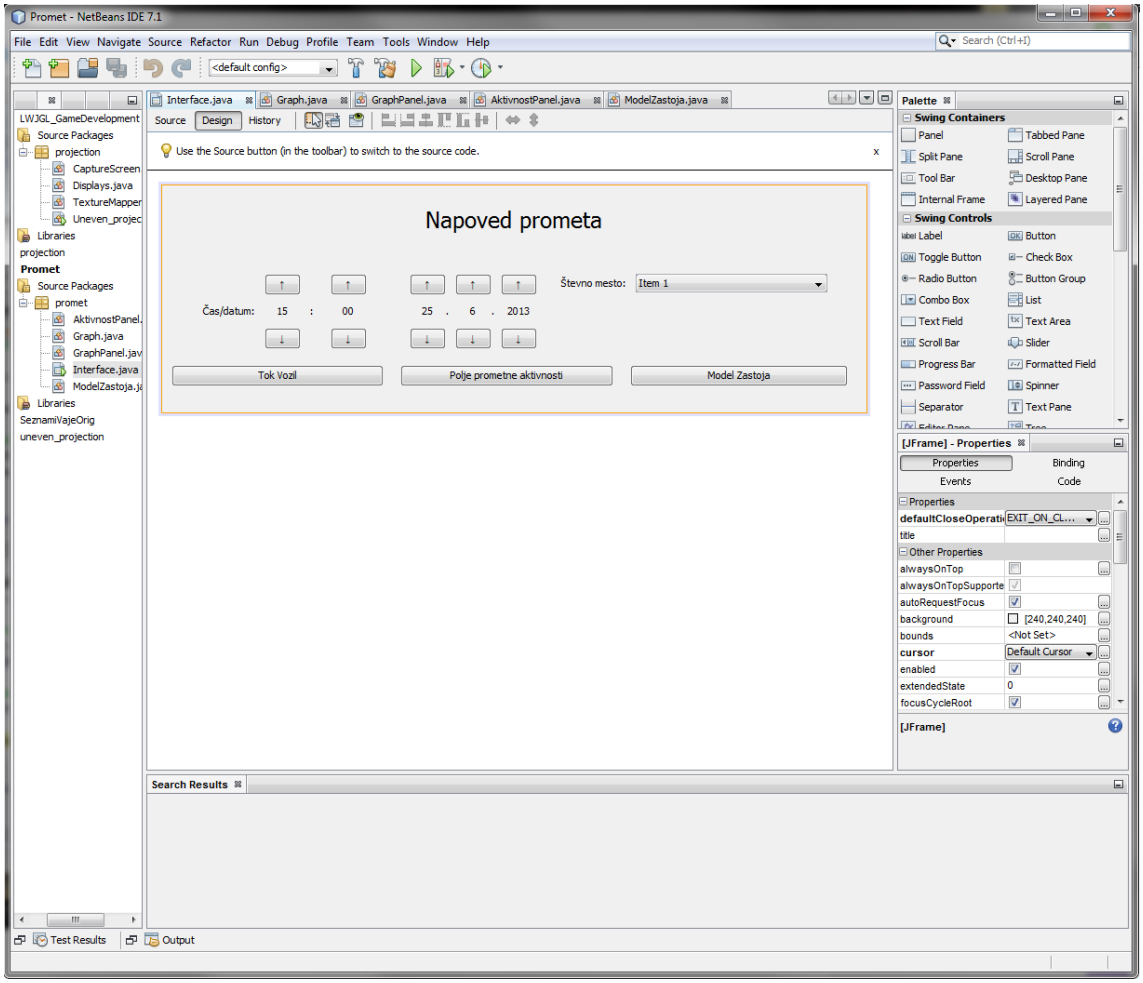
2.2 IDE

Integrirano razvojno okolje (angl. Integrated development environment) je aplikacija, ki vključuje orodja za lažji razvoj programske opreme. Običajno je sestavljen, iz urejevalnika izvorne kode, razhroščevalnika in prevajalnika. Včasih so v orodje vgrajeni tudi sistem za nadzor verzij programov in predvsem orodja za tvorbo grafičnih uporabniških vmesnikov bodočega programa. Mnoga sodobna okolja imajo tudi orodja za pametno dokončevanje kode.

Integrirana razvojna okolja so namenjena optimiziranju programerjeve produktivnosti. Nekatera okolja so namenjena specifičnemu programskemu jeziku, spet druga omogočajo programiranje v večih programskih jezikih.

2.3 NetBeans

NetBeans je platforma za razvoj programske opreme, narejena v Javi. Zaradi modularne zasnove omogoča preprosto razširitev uporabnosti. Integrirano programsko okolje NetBeans, ki temelji na platformi NetBeans, je primarno namenjeno razvoju v Javi, vendar podpira več jezikov (PHP, C/C++, HTML5, ipd.). V osnovni različici nam ponuja poleg osnovnih orodij, ki so prisotna v razvojnih okoljih, tudi orodje za oblikovanje uporabniških grafičnih vmesnikov.



Slika 2.2: Izgled orodja NetBeans za oblikovanje uporabniških grafičnih vmesnikov.

Poglavje 3

Napovedovanje prometa

Zanimanje za pojem prometne dinamike je presenetljivo staro. Že v letu 1935 je Greenshields izvedel študije o prometu in že v 50ih smo s pomočjo raziskav dokumentirali pojme kot so temeljni diagram in nestabilnost pretoka, ki sta še vedno pomembna. Vsa ta leta kasneje pa se je situacija samo poslabšala. Z večjim številom vozil na cestah danes v večjih mestih vozniki zapravljajo čas čakajoč v zastojih, kar znese nekaj dni na leto. Ob počitniških sezonah se pojavijo zastoji dolgi več 100 kilometrov. Emisije vozil, kot so SO_2, CO_2, hlapi in hrup so dosegle in v nekaterih primerih celo presegle nivoje gospodinjstev ali celo industrijskih obratov. Zaradi tega je potreba po učinkovitem prevoznem sistemu je brezdvomno bistvena za uspeh moderne industrijske družbe.

Napovedovanje prometa se nanaša na ocenjevanje števila vozil v prihodnosti. Napovedovanje se začne z zbiranjem podatkov o trenutnem prometu. Ti podatki se nato združijo z ostalimi znanimi podatki (npr. število prebivalstva v okolici, potni stroški,...), kar nam omogoči oceniti promet za izbran čas t in prostor x. Poznamo makroskopski in mikroskopski pristop k modeliranju pretoka prometa. Pri mikroskopskemu modelu simuliramo posamezna vozila, zato dinamične spremenljivke predstavljajo mikroskopske lastnosti, kot so lokacija in hitrost posameznih vozil. Tak pristop je običajno prezapleten za spletne aplikacije, kjer so iskane bolj splošne lastnosti prometa.

Posledično se odločimo za makroskopski pristop, ki temelji na dveh spremenljivkah. Povprečna hitsrost V in gostota vozil p (Kerner, 2001). Z makroskopskim modelom promet opredeljujemo kot celoto namesto opisovanja lastnosti posameznih vozil.

3.1 Meritvene tehnike

Najbolj uporabljena tehnika merjenja prometa je verjetno zračno fotografiranje prometa, saj nam omogoča beleženje velikega števila avtomobilov, njihovo smer in hitrost.
Spet druga metoda bi lahko uporabljala podatke novejših avtomobilov, ki lahko posredujejo svojo lokacijo, hitrost, pospešek, mogoče celo razdaljo med avtom pred njimi. Vendar je večina podatkov o avtomobilih pridobljena s strani avtomatskih števcev postavljenih na določenih mestih avtoceste. Števci lahko beležijo število vozil v določenem časovnem intervalu, kot tudi čas prihoda ter odhoda vozila iz dosega detektorja. Novejši števci pa lahko beležijo še hitrosti vozil in njihove dolžine, kar nam omogoča oceniti razdalje med vozili.

3.2 Temeljni diagram

Po dobljeni informaciji o izbranemu števnemu mestu in datumu napovedi aplikacija, s pomočjo te informacije, izlušči samo podatke, potrebne za napoved izbranega dne. Razmerje med pretokom prometa, povprečno hitrostjo in gostoto prometa merimo že desetletja (Greenshields 1935), kjer smo našli linearno razmerje. Izraz temeljni diagram se nanaša na funkcijo

$$Q = pV, \tag{3.1}$$

kjer so:

- Q - pretok prometa, število avtomobilov na uro $\frac{N}{h}$,

- p gostota prometa, število avtomobilov na kilometer $\frac{N}{km}$,
- V povprečna hitrost, kilometri na uro $\frac{km}{h}$.

Za naš primer bomo zaradi oblike podatkov in ravnovesja, ki ga vidimo v formuli (3.3), temeljni diagram opremili z časovno (t) in prostorsko (x) spremenljivko.

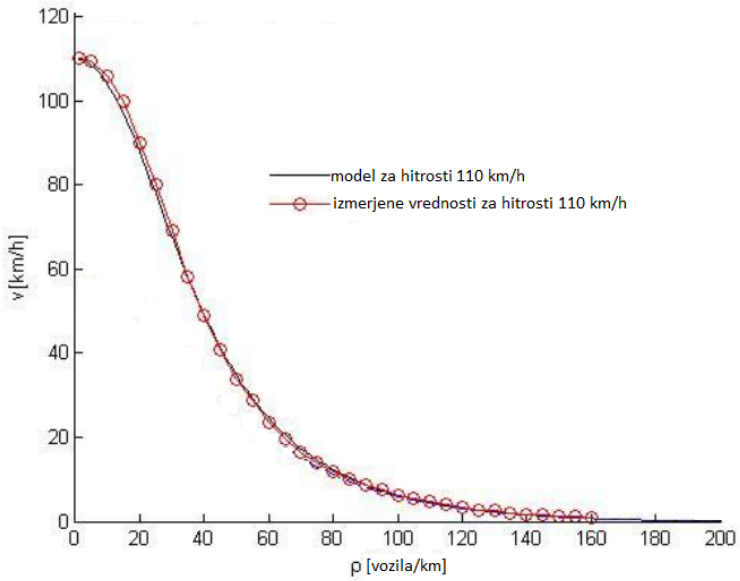
$$Q(x,t) = p(x,t)V(x,t). \tag{3.2}$$

Pri napovedovanju prometa smo se odločili za makroskopski opis prometa, ki temelji na dveh spremenljivkah. Povprečna gostota prometa p in hitrost avtomobilov V (Kerner, 2001; Treiber and Kesting 2010). Pri makroskopskemu modelu se omejimo na opis kolektivne dinamike vozil, v smislu prostorske gostote $p(x,t)$ in povprečne hitrosti $V(x,t)$, za podan prostor x in čas t. Delovanje temelji na dejstvu, da se število vozil na avtocesti ne spreminja (Lighthill, Whitham, 1955). To ravnovesje vozil lahko prikažemo z

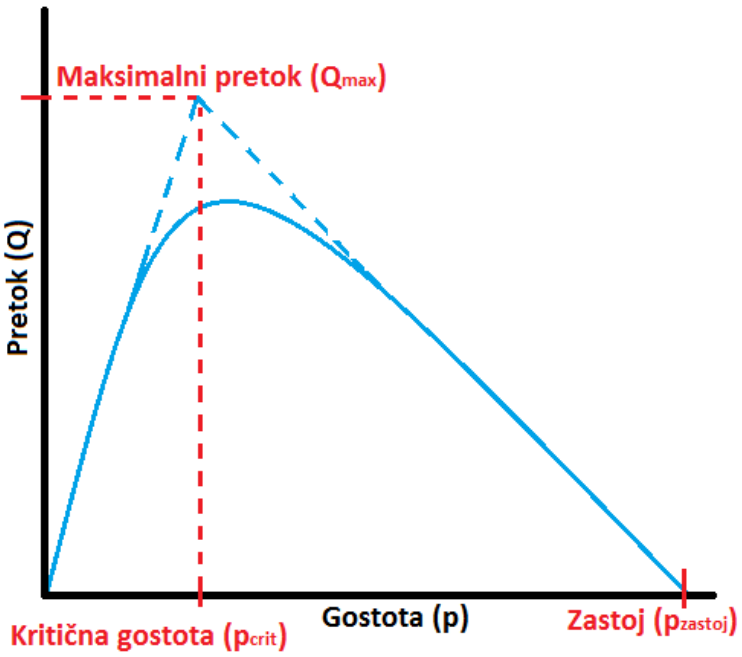
$$\frac{\partial p(x,t)}{dt} + \frac{\partial Q(x,t)}{\partial x} = 0. \tag{3.3}$$

Večina meritev potrdi slednje lastnosti:

- Pri nizkih gostotah p je razmerje med gostoto in pretokom enodimenzionalno. Začne se linearno in je ukrivljeno navzdol.
- Z naraščanjem gostote prometa se hitrost niža in izgine skupaj z pretokom v neki maksimalni gostoti ceste p_{zastoj} (slika 3.1), kjer nastane zastoj.
- Pretok prometa ima nek maksimum Q_{max} pri srednjih gostotah.
- Kot vidimo v sliki 3.2, razmerje med pretokom in gostoto izgleda kot zrcalna oblika grške črke lambda (Kerner, 2000a).



Slika 3.1: Razmerje med hitrostjo in gostoto. Podatki so vzeti iz reference (Helbling, 1997).



Slika 3.2: Razmerje med pretokom in gostoto. Vidimo da se z večanjem gostote pretok počasi niža, dokler v neki točki p_{zastoj} ne izgine. Rdeči črti prikazujeta kritično gostoto in maksimalni pretok.

Poglavje 4

Aplikacija za napovedovanje prometa

4.1 Ideja

Ideja za aplikacijo za napovedovanje prometa je nastala iz vprašanja, kako bi lahko zmanjšali čas čakanja v zastojih, ter kako bi voznikom dali potrebno znanje, da se zastojem izognejo.

Idejni načrt:

- Prikažemo uporabniški vmesnik.
- Preberemo izbrano števno mesto in datum.
- Izvlečemo potrebne podatke za izbran datum.
- Interpretiramo podatke in izrišemo graf.

Glavna ideja je izdelati aplikacijo, ki bi uporabniku omogočala učinkovito odločitev o času odhoda.

4.2 Funkcionalne zahteve

Natančnejše specifične funkcionalne zahteve so:

- Želimo razviti aplikacijo za napovedovanje prometa.
- Aplikacijo naj bo možno uporabljati na osebnih računalnikih in prenosnikih, ki imajo nameščenega vsaj enega izmed operacijskih sistemov Microsoft Windows, Linux, Mac.
- Aplikacija mora omogočati izbiro števnega mesta in datuma napovedi.
- Aplikacija mora napovedati predviden čas zastoja.
- Aplikacija mora napovedati predvideno število vozil v zastoju.

4.3 Predstavitev aplikacije

V tem delu si bomo ogledali uporabnikov nadzor nad aplikacijo in njenim delovanjem. Aplikacija najprej prebere podatke od števnih mestih, ter prikaže uporabniški vmesnik za izbiro števnega mesta in datuma napovedi. Podatke o pretoku prometa imamo v obliki število avtomobilov n na uro t za števna mesta in vse dni v letu. Aplikacija uporabi izbrano števno mesto in datum, da izlušči samo potrebnih 24 ur in jih shrani v obliki List.

```
public List<Double> PROMET = new ArrayList<>();
```

4.3.1 Grafični uporabniški vmesnik

S pomočjo grafičnega uporabniškega vmesnika lahko uporabnik izbere lokacijo in čas izračuna modela. Kot vidimi na sliki (4.1) uporabnik izbere parametre s pomočjo gumbov, kar nam omogoča nadzor nad napačnimi vnosi. Uporabnik nato izbere model napovedi.

Oblika podatkov posameznih števnih mest:

Slika 4.1: Grafični uporabniški vmesnik, ki omogoča izbiro števnega mesta in datuma napovedi.

- Števno mesto: Identifikacijska številka števnega mesta.
- Številka ceste: Identifikacijska številka ceste.
- Številka odseka: Identifikacijska številka odseka ceste, ki ga pokriva števno mesto, saj imajo nekatere ceste več števnih mest.
- Ime števnega mesta: na primer "Lom AC".
- Tip štetja: Tip avtomatkega štetja ali oznaka za ročno štetje vozil.
  - QLD3: prešteje vsa vozila.
  - QLD5: loči 5 kategorij vozil.
  - QLD6: loči 10 kategorij vozil.
  - QLTC8: loči 10 kategorij vozil.
  - QLTC10: loči 10 kategorij vozil.
  - QLD: štetje z različnimi tipi števcev.
  - ROČNO: ročno štetje vozil.
- Smer: smer vožnje prometa, v katero se šteje.
- GKX: X Gauss-Krieger koordinata števnega mesta.

- GKY: Y Gauss-Krieger koordinata številnega mesta.

4.3.2 Funkcija glajenja

Ker imamo podatke o številu vozil ob vsaki uri v dnevu, bomo dobili zelo oster graf (npr. slika 4.2). Zaradi tega podatke zgladimo z funkcijo

$$Q_i(t_j) = \frac{\sum_{n=0}^{24} tok_n e^{-\frac{1}{2}(\frac{t_j - n}{\rho})^2}}{\sum_{n=0}^{24} e^{-\frac{1}{2}(\frac{t_j - n}{\rho})^2}}, \tag{4.1}$$

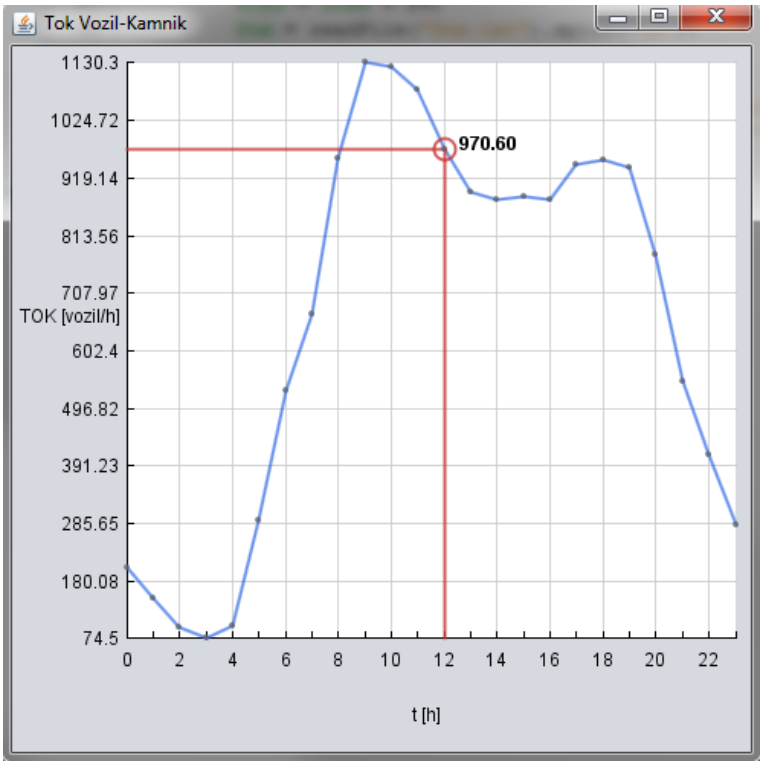
ki pa je v kodi izvedena kot:

```
for(int i = 0; i < smax; i++){  //smax - (1, 2,...., 1440).
   double pow = 0, exp = 0, helpE = 0;
   double pomF1 = 0, pomF2 = 0;
   for(int j = 0; j < 24; j++){
      helpE = ((t[i]-time[j])); //t[i] - (1/60, 2/60,..., 1440/60).
      pow = -(Math.pow(helpE, 2));//time[j] - (1, 2,..., 24).
      exp = Math.exp(pow/sig2); //sig2 - parameter v gausovki (0.5).
      pomF1 = pomF1 + (tok[j] * exp);
      pomF2 = pomF2 + exp;
   }
   F1[i] = pomF1;          //F1 in F2: pomozni spremenljivki.
   F2[i] = pomF2;
   qi[i] = F1[i]/F2[i];
}
```

S tem dosežemo boljšo aproksimacijo prometa na izbran dan, saj promet celotnega dneva vpliva na vsako izbrano minuto.

4.3.3 Tok vozil

Za napoved toka vozil, preberemo podatke v seznamu in jih izrišemo na graf, ki ga vidimo na sliki 4.2. Graf je sestavljen iz povezanih točk, ki ponazarjajo



Slika 4.2: Graf toka vozil, izrisan po podatkih števnih mest. Število vozil ob izbrani uri je prikazano z rdečo črto.

število vozil v vsaki uri.

4.3.4 Model zastoja

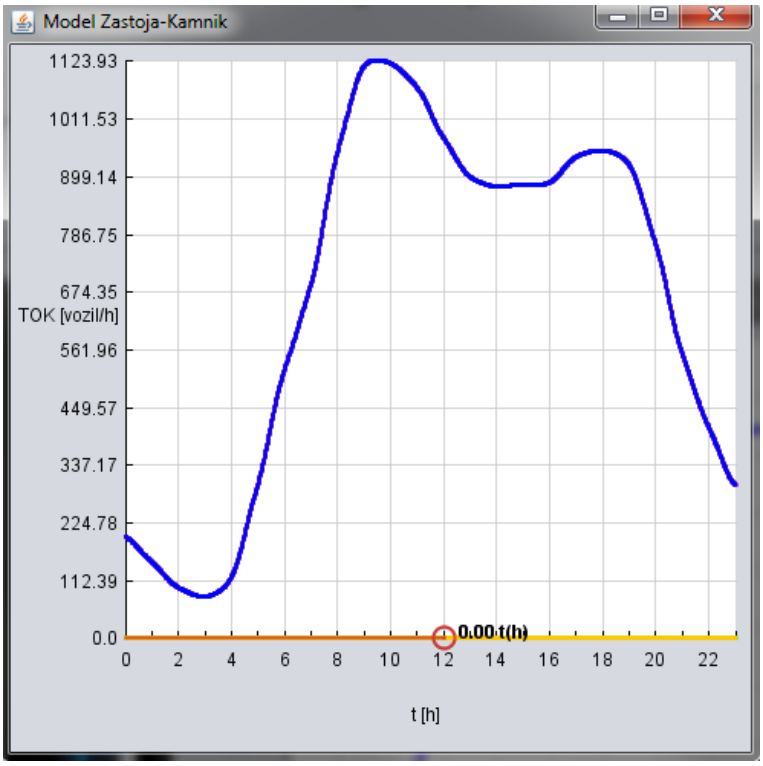
Model napove zastoj predvidenega toka vozil. V prvem koraku s pomočjo funkcije glajenja, funkcija 4.1 na strani 18, iz grafa toka vozil, dobimo graf modela zastoja, ki ga vidimo na sliki 4.3. Model nato primerja predvideno število vozil z kapaciteto ceste. V primeru, da količina vozil presega maksimalni pretok ceste vemo, se pojavi zastoj. V grafu (slika 4.3) to vidimo v točki, kjer modri graf neha naraščati in se omeji na kapaciteto ceste. Kapaciteta ceste je določena Določitev števila vozil v zastoju in časovno zakasnitev izračunamo na slednji način

```
for(int i = 0; i < (I-1); i++){
   if(z[i]>0){
      qo[i]=qzmax;
   }
   else{
      if(qi[i]<=qzmax)
         qo[i]=qi[i];
      else
         qo[i]=qzmax;
   }
   z[i+1]=z[i]+(qi[i]-qo[i])*dt; //stevilo vozil - N
}
for(int i = 0; i < pomZ.length; i++){
   pomZ[i]=z[i]/qzmax;        //cas t[h]
}
```

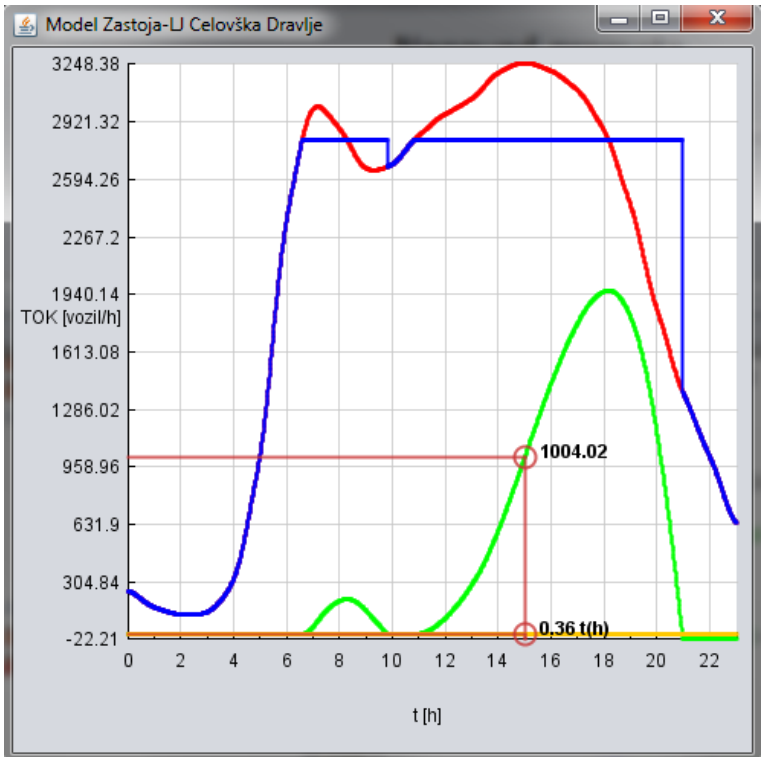
Na sliki 4.3 vidimo primer, ko število vozil ne presega zmogljivosti ceste. Vidimo prost pretok prometa označen z modro črto. V naslednjem primeru 4.4 vidimo, da se dejanski pretok (modri graf) pri neki točki ustavi, dokler se število vozil v zastoju ne zniža dovolj, da se pretok spet odpre. Zeleni graf predstavlja število vozil v zastoju, medtem ko je rumeni graf predviden čas zastoja. Z rdečima črtama je prikazano število vozil in čas zastoja ob izbrani uri. Rdeči graf pa predstavlja pretok prometa, če do zastoja ne bi prišlo. Za primerjavo lahko na sliki 4.5 vidimo predviden promet za isto števno mesto nekaj dni kasneje v nedeljo. Opazimo, da se promet ob vikendih znatno zniža.

4.3.5 Polje prometne aktivnosti

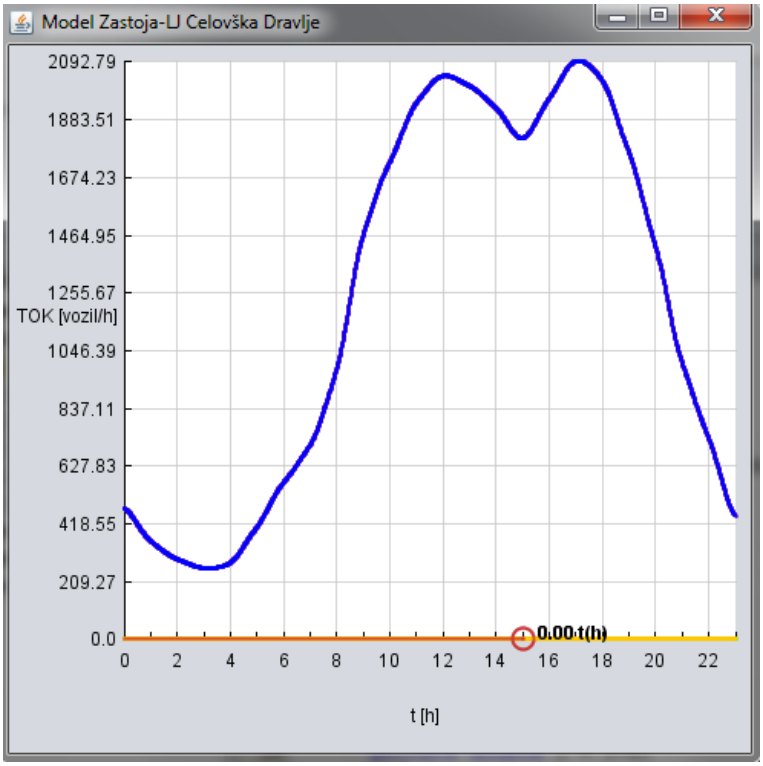
Polje prometne aktivnosti prebere podatke izbranega dne in ure in za vsa števna mesta predvidi količino vozil. Nato s pomočjo lokacij števnih mest izriše promet na avtocestah za celotno Slovenijo. Na sliki 4.6 vidimo stanje prometa v Sloveniji, pri čemer so bolj prometna območja označena z rdečo



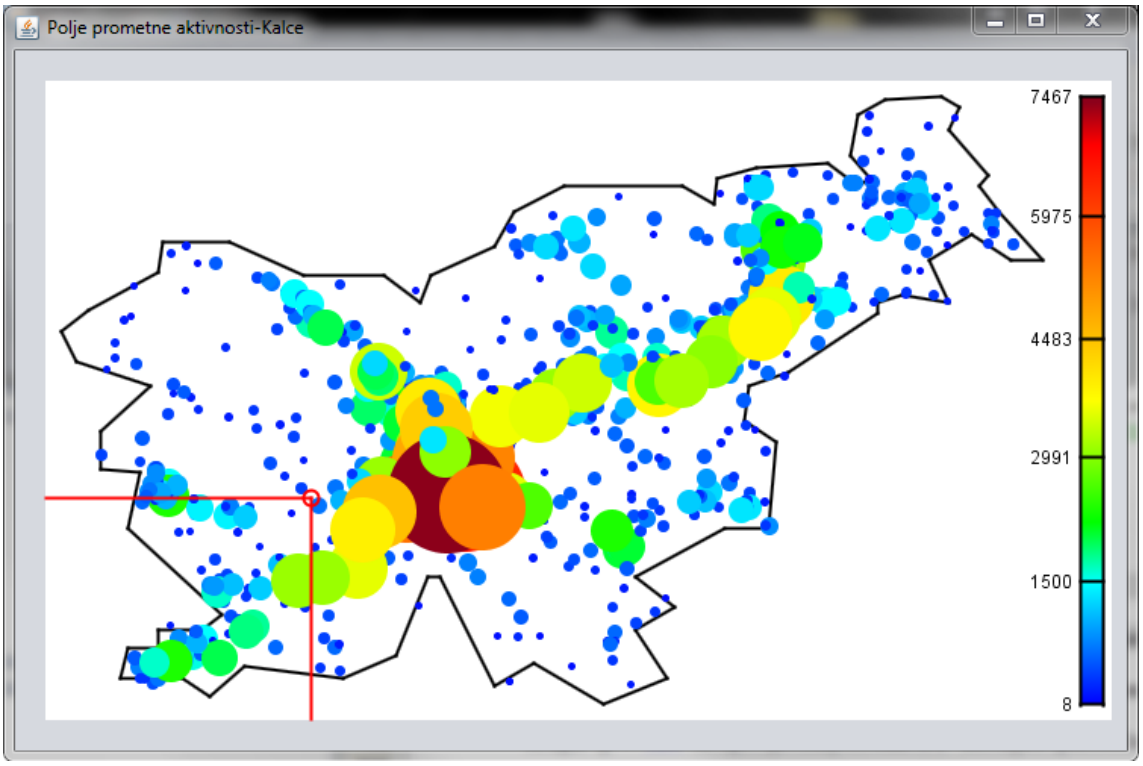
Slika 4.3: Predviden tok prometa na števnem mestu "Kamnik"v sredo 23. septembra ob 12:00.



Slika 4.4: Model zastoja, ko je število vozil preseglo zmogljivost ceste. Tok je predviden za števno mesto "LJ Celovška Dravlje"v četrtek 24. septembra ob 15:00.

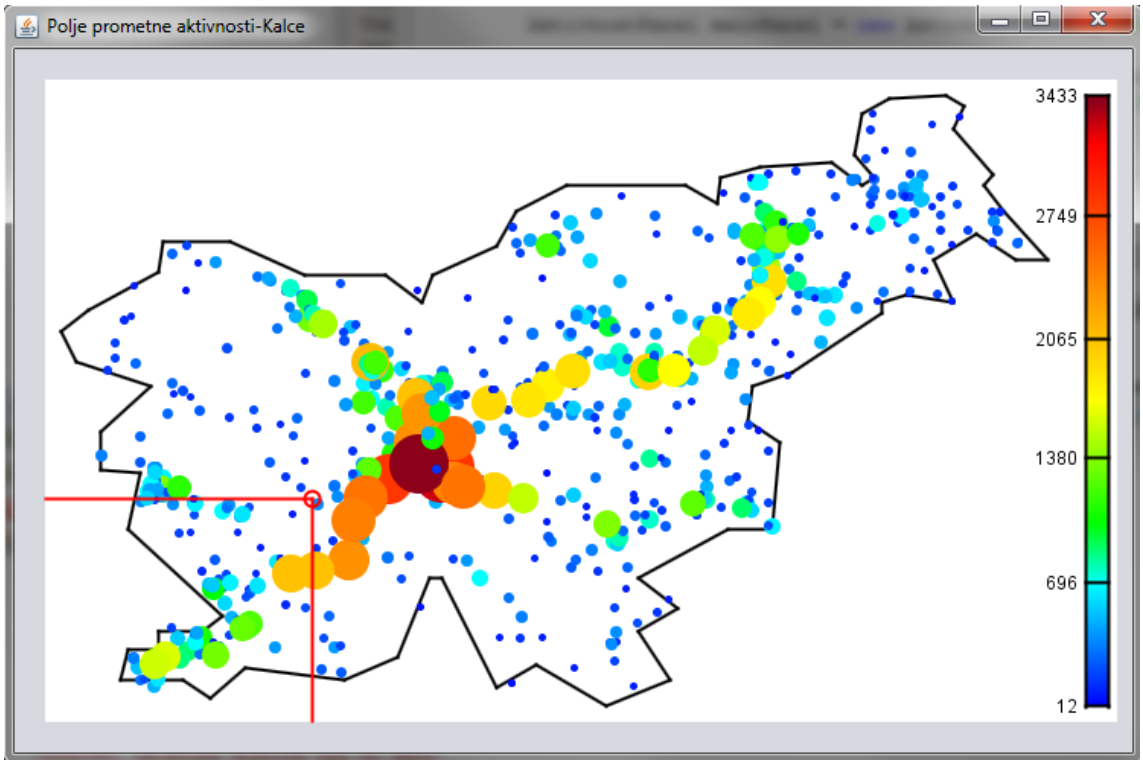


Slika 4.5: Predviden tok prometa na števnem mestu "LJ Celovška Dravlje" v nedeljo 27. septembra ob 15:00.



Slika 4.6: Polje prometa ob delovniku.

barvo in večjimi krogi, manj prometna pa z modro barvo in manjšimi krogi. Rdeči črti prikazujejo lokacijo izbranega številnega mesta. Na sliki 4.7 pa vidimo, da se ob vikendih promet umiri po celi Sloveniji.



Slika 4.7: Polje prometa ob vikendih.

Poglavje 5

Sklepne ugotovitve

Študija pretoka prometa je zapleten primer, saj je voznikovo vodenje nemogoče predvideti z stoodstotno gotovostjo. Na srečo se večina voznikov drži pravil, kar nam omogoča razvoj matematičnega modela z razumno gotovostjo. Pokazali smo, da kljub zapletenosti prometa lahko, s pomočjo temeljnega diagrama (Treiber, Kesting, 2010), modeliramo lastnosti prometa in s tem modeliramo ravnovesje prometa. Za razvoj našega modela smo uporabili relativno preprost opis prometa, vendar smo z njegovo preprosto zasnovo, pripravljeni na razširitve. Model lahko preprosto razširimo, z upoštevanjem del na cesti, ki so območja drastično znižane povprečne hitrosti. Taka območja bi potem bolj prispevala k razširjanju prometnega zastoja.

Literatura

[1] (2015) NetBeansIDE. Dostopno na:
`https://netbeans.org/features/index.html`

[2] (2015) Oracle "Java Overview". Dostopno na:
`https://www.oracle.com/java/index.html`

[3] (2015) Napovedovanje prometa. Dostopno na:
`https://en.wikipedia.org/wiki/Transportation_forecasting`

[4] Grabec, I., Kalcher, K., Švegl, F.(2010):
Modeling and forecasting of traffic flow.

[5] Helbing, D. (1997): Verkehrsdynamik. Berlin: Springer.

[6] Kerner, B.S. (2004): The Physics of Traffic. Berlin: Springer.

[7] Treiber, M., Kesting, A. (2010): Verkerhsdynamik und Simulation. Berlin: Springer.